The Influence of Epistemology on the Design of Artificial Agents

Lee, Mark; Lacey, Nick

Published in:
Minds and Machines

DOI:
10.1023/A:1024197120231

Publication date:
2003

Citation for published version (APA):
Lee, M., & Lacey, N. (2003). The Influence of Epistemology on the Design of Artificial Agents. *Minds and Machines*, *13*(3), 367-395. https://doi.org/10.1023/A:1024197120231

The Influence of Epistemology on the Design of Artificial Agents

M. H. Lee
Centre for Advanced Software and Intelligent Systems,
Department of Computer Science,
University of Wales, Aberystwyth, SY23 3BD,
Wales, UK.

and

N. J. Lacey
Multi-Agent Systems Research Group,
Department of Computer Science and Computer Engineering,
University of Arkansas, Fayetteville, Arkansas 72701,
USA

January 14, 2008

Abstract

Unlike natural agents, artificial agents are, to varying extent, designed according to sets of principles or assumptions. We argue that the designer's philosophical position on truth, belief and knowledge has far reaching implications for the design and performance of the resulting agents. Of the many sources of design information and background we believe philosophical theories are under-rated as valuable influences on the design process. To explore this idea we have implemented some computer-based agents with their control algorithms inspired by two strongly contrasting philosophical positions. A series of experiments on these agents shows that, despite having common tasks and goals, the behaviour of the agents is markedly different and this can be attributed to their individual approaches to belief and knowledge. We discuss these findings and their support for the view that epistemological theories have a particular relevance for artificial agent design.

Keywords: Autonomous agents, Agent design philosophy, Agent knowledge bases, Relations between Philosophy and Artificial Intelligence.

1 Introduction

Autonomous cognitive agents, whether natural or artificial, are information processing entities that make decisions, recognise patterns, gather information and perform actions. The concept of autonomy refers to the ability to use experience to determine action. This includes being able to adapt behaviour in order to pursue goals under changing circumstances. Artificial agents can take a range of forms from software agents to anthropomorphic robots. They have been characterised according to their degree of participation in their environment, with the most highly integrated being described as situated, embedded or embodied [3]. A situated agent is one that has a direct coupling with its environment, rather than through some intermediate representation; while an embedded agent has a stronger integration where the environment is a necessary component in shaping the agent's behaviour and goals. Most conventional computer programs are not agents because their coupling with an environment is very weak, although they may have significant indirect effects (e.g. weather prediction). Examples of situated agents include the software agents (softbots) that are used in personalised web scanning systems which select or analyse specific material for their customers. Examples of embedded agents include auto-pilots and other sophisticated feedback control systems. Embodiment refers to the agent having a physical structure within the environment itself, which necessarily gives the potential to influence or disturb situations and events. Perhaps robots (notwithstanding humans) are the best examples of autonomous agents that are situated, embedded and embodied. As our main concern is with the relationships between agent and environment we will use autonomous mobile robots as our case study system throughout, (and will use the terms agent and robot synonymously). This will ensure a close coupling between agent and environment and emphasise the complexity of real world applications.

We will consider an agent to be an embedded computer system that constructs and manipulates an internal world model on the basis of sensory information. This internal model will be held as a representation in some form of knowledge-base that we will call an *Agent Knowledge-Base* (AKB). By existing and reacting in its domain, an agent will build and maintain an AKB that represents what it knows to be true. The approach an agent takes to truth depends upon which theory of truth is upheld, which in turn depends upon the underlying metaphysical foundations of the agent. For example, one agent might be designed according to a theory of objective truth based on correspondence, whereas another agent could be founded on a different theory based upon coherence. However, even if an agent's designer is not aware of such distinctions, knowledge processing design decisions will be made that *implicitly* represent some form of metaphysical or epistemological position.

In a previous paper, we discussed the metaphysical theories of realism and anti-realism. We argued that, although being abstract, such concepts were nevertheless relevant to the various decisions that are made when an AKB is being designed [5]. In particular, different epistemological theories will have significant effects, and these should be manifest when agents based on different views of truth, belief and knowledge produce different behaviours and performances. In order to test this idea we have implemented several agents with deliberately different stances on these issues. The agents all have the same goals, environment and physical structure but their control algorithms are developed from opposing epistemological positions. The purpose of these implementations is to firmly ground the theories and concepts in testable realisations so that we may examine if and how philosophical design differences between the agents will lead to significant differences in their behaviour.

2 The Design of Two Agents

We now develop two strongly contrasting frameworks for two different agents. The aim is to deliberately chose two very different positions on belief and knowledge so that any differences in performance on a common task will be clearly exaggerated to assist analysis. The two designs are based on a strongly holistic approach and a strongly atomistic approach. We also find it instructive to introduce a third design as an intermediate stage.

Our focus is on robotic agents and so the inputs will be sensory data and the outputs will be actions applied to the environment. Because the environment, physical structure and task (simple navigation) are all fixed, much of the design, for any agent, may be thought to be already strongly determined by the functional aspects required to achieve the task. However, there are still ontological choices open for the agents, and these will colour their designs. The first choice is the nature of sensed information and its status as regards experience. A second choice is a theory about truth, and a third concerns how beliefs are justified.

2.1 A Strongly Atomistic Position - PA

Position **PA** represents a view in which packets of incoming information are treated as atomic, independent facts that have high intrinsic value. Each piece of sensory data is seen as carrying its own meaning about the state of the world, independently of any other considerations. In this form of fairly extreme atomism, previous experience has no role to play in interpreting sensory data.

An appropriate theory of truth for atomism is a correspondence theory in which a belief is accepted as true if it describes an actual state of the world. That is, sensory data that are believed are considered to correspond to true facts about the world.

Belief justification is the process by which a belief is supported or gains support, usually from other beliefs. If each atom of incoming sensory data is to carry its own meaning, then it can not be justified by other beliefs. However, foundationalist theories of justification allow the use of a special class of beliefs, *basic beliefs* which are never justified themselves but can justify other beliefs [2]. In this way empirical evidence can be treated as basic beliefs and then support other, inferential, beliefs.

2.2 A Strongly Holistic Position - PH

In contrast to **PA**, position **PH** has been designed to be based primarily on holistic principles. A holistic view sees it important to reconcile new sensory data with existing experience. New information must be evaluated in terms of what is already known, and in strong holism this means being assessed against *all* relevant knowledge.

Our strongly holistic position will adopt a coherence theory of truth, that is, a belief will be considered true if it is compatible with an existing set of (coherent) beliefs.

Regarding justification, all beliefs will be involved in mutual support. The degree of justification of a belief relates to the consistency and compatibility of that belief with the belief set of which it is a member.

2.3 The Philosophical Differences

We can view these positions as offering a set of options available to AKB designers that lie on a kind of spectrum. Figure 1, adapted from [7], illustrates this idea. The shaded areas represent the extreme positions represented by **PA** and **PH**.

In approach **PA** sentences contain their own atomic meaning. An approach to truth based on **PA** will be based on a correspondence theory of truth and a foundationalist theory of justification will be used.

An approach based on **PH** will be based on coherence, rather than correspondence. Under this view, the truth of a belief is determined in terms of its relationship with an agent's existing beliefs. The approach to justification based on **PH** is also based on coherence, in that a belief is

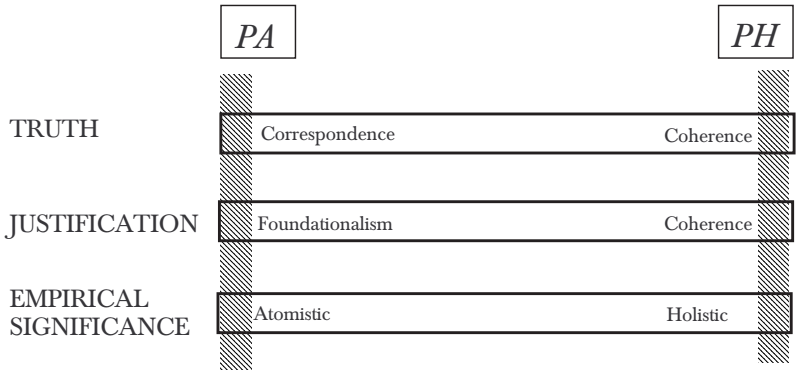

Figure 1: AKB Design Options at the Philosophical Level

justified to the extent that it is a member of a coherent set of beliefs. Thus, no belief is immune from revision.

We summarise the different metaphysical and epistemological positions for the two agents in Table 1. For further elaboration of these issues, see [5].

Concept	**Position PA**	**Position PH**
The nature of the external world	Objectively true independently of verification	Truth depends on context and is dynamic
The unit of empirical significance	Individual beliefs and perceptions	The entire knowledge base
Truth	Defined in terms of correspondence with the external world	Defined in terms of coherence with an agent's existing beliefs
Justification	A foundationalist approach based on basic beliefs which require no justification	A coherence-based approach in which *every* belief may be open to revision

Table 1: The Differences Between Approaches **PA** and **PH**

2.4 The Functional Requirements of the Agents

We now develop the requirements and features that agents based on **PA** or **PH** will need if they are to be fully realised as implementations. The functionality of the agents will be described in terms of three key processing stages: **Data Input** (from the external world), **Derivation** (combining new beliefs with existing beliefs), and **Evaluation** (knowledge base revision).

It is useful to identify three classes of beliefs: **Basic Beliefs** – B_K, as used by the the agents based on **PA**, **Assumed Beliefs** – A_K which have been built into the system by the system designer, and **Derived Beliefs** – D_K which have been derived from basic or assumed beliefs.

2.4.1 Agents Based on PA

We have developed two variants based on position **PA**: agent **SA** is strongly based on **PA**, while agent **WA** is based on a weaker form of **PA**.

First we consider **Data Input**. The atomistic agents define accuracy in terms of correspondence with the external world. This means that information from the agent's senses will be treated with utmost importance and an agent based on **PA** would be expected to label each piece of data with a binary truth indicator as soon as it enters the system without any form of pre-processing. Agent **WA**, however, relaxes this slightly in that it is able to make decisions on the basis of several consecutive pieces of sensory data.

Regarding **Derivation**, an agent based on **PA** would derive all its information concerning the state of the external world from its senses. This derivation process would take the form of justification based on foundationalism, whereby the agent's senses provide the basic beliefs B_K from which the rest of the agent's beliefs D_K are derived. Thus, in these agents, basic beliefs are not themselves inferentially justified, and require no justification. There is no **Evaluation** stage for these agents as their knowledge bases are not subject to revision.

The Strongly Atomistic Agent - SA

Each sensory unit of data in agent **SA** is treated separately as having its own value and meaning. These sensory data form its basic beliefs B_K and the rest of its beliefs, D_K, are then (asymmetrically) derived from B_K. However, we notice that this approach to agent design poses certain problems. Because all sensory data are of high importance, they must be dealt with immediately and any consequent changes will be incorporated in the internal model. Thus, agent **SA** has no ability to amend the value of its sensor data and if these data are not entirely accurate they will have the potential to degrade the quality of its internal model. Thus, the performance of agent **SA** with imperfect sensors or perceived uncertainties in the environment is expected to be disappointing.

The Weakly Atomistic Agent - WA

While minimising any shift away from position **PA**, agent **WA** has been designed to address the problems caused by inaccurate sensor data. This has been done using a system of *suggestions*. Whereas agent **SA** acts immediately on new sensor information, agent **WA** translates the sensor information it receives into suggestions which may be acted upon if they receive enough support. This has the effect of averaging sensor data over several consecutive sensory readings. The number

over which sensor data should be averaged, as well as the amount of support which suggestions require before being carried out, can both be set as parameters.

Even though some theoretical purity has been sacrificed at the expense of operational improvements, agent **WA** can still claim to be based on **PA**. This is because (a) all the information used by agent **WA** is derived from its sensors, and (b) the process by which the agent decides which suggestions to act on is atomistic, being based on individual beliefs.

2.4.2 An Agent Based on PH

Agent **SH** is a strongly holistic design where the accuracy of belief ϕ depends on the extent to which ϕ coheres with the agent's other existing beliefs. For this reason, agent **SH** should have the ability to analyse the data it receives, and then accept it, modify it, or even reject it altogether, depending on which course of action leads to the most coherent knowledge base.

Regarding **Data Input**, agent **SH** has the power to reject sensor data, and so great care must be taken to ensure that agent **SH** does place sufficient importance on sensor data. Coherence-based epistemological theories are often criticised for their inability to explain the importance we attach to empirical information. How can agent **SH** avoid becoming a target of similar criticism?

This criticism would be justified if agent **SH** were to *ignore* sensor information. However, rather than being ignored, incoherent sensor data are *rejected*. The difference is that to ignore ϕ is to behave as if ϕ never happened, while to reject ϕ is to decide, on the basis of existing beliefs, that ϕ is inaccurate. In order for agent **SH** to decide that rejecting ϕ is the best course of action, the resulting knowledge base $K^{-}\phi$, *including* the explanation of the rejection of ϕ, must be more coherent than $K^{+}\phi$. Thus, the process of entering data into K is concerned with maximising the coherence of K.

Regarding **Derivation**, the method used by agent **SH** to derive its knowledge base is based on the coherence theory of justification. Figure 2 reveals several important differences between the approach to justification taken by **PA** and **PH**. Firstly, the coherence approach to justification means that *all* beliefs are inferentially justified, and that even the most central beliefs can be in part justified by less central beliefs. Secondly, agent **SH**'s beliefs are organised in a web-like manner. Beliefs which are central to the agent's belief system are stored at the center of this web, while those which are less important are stored at the periphery. Empirical data, while providing most of the agent's information concerning the external world, are not central to the agent's belief structure. This non-central positioning of empirical data allows agent **SH** to amend its sensor data when necessary.

When required to maximise the coherence of K, we use the term "integrity" as a measure for comparing the coherence of knowledge bases in **SH** agents. Our method used to calculate the integrity of a knowledge base $I(K)$ is described in detail in [4] and includes a mixture of all of the following concepts: consistency, mutual belief support, explanatory power, and the principle of minimal change.

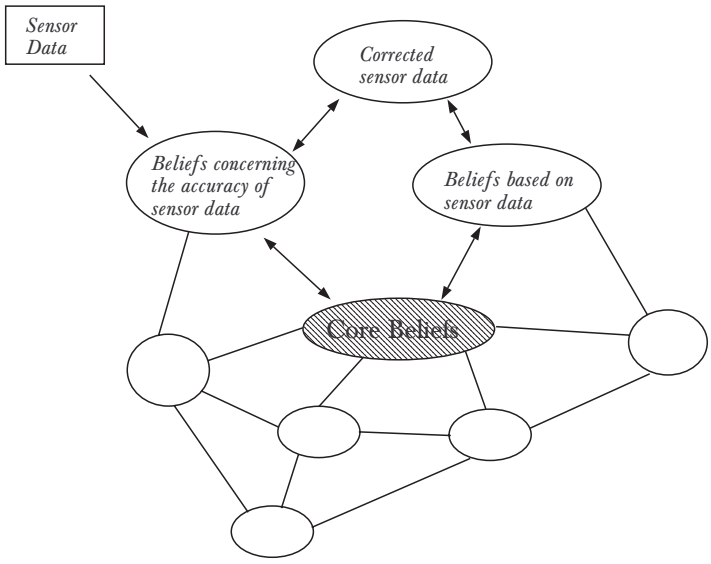

Figure 2: The Approach to Justification Based on **PH**

Evaluation for agent **SH** is the process whereby the integrity of a derived knowledge base is calculated, any problems are identified, and possible solutions, in the form of changes to the knowledge base, are suggested. Once all the possible solutions have been derived, the integrity of each one is calculated, and the solution with the highest integrity is adopted. These solutions are obtained independently, meaning that if n solutions are to be evaluated, the values of $I(K_1), I(K_2), \ldots, I(K_n)$ will have to be obtained. The fact that **PH** prohibits the use of a pseudo-objective standpoint from which to obtain these values means that all integrity measures must be derived from the point of view of the base interpretation K_b.

3 Agent Implementations

In order to provide a realistic test in a concrete application domain, the three agents described above were implemented as autonomous mobile robots that must navigate through their environment and reach a goal location. A simple simulated robot body and environment was used which gives the advantage that the problem and environment can be kept identical for all agents but different control algorithms can realise any theoretical differences originating from their contrasting philosophical approaches. We are aware of, and agree with, the argument that simulation is not a sound methodology for representing and analysing all the nuances of real applications [6], but it is well suited for design analysis and other high level studies of control regimes and their differences. In this case, the control, consistency and repeatability of the simulation tool gives an ideal scientific scenario for comparative experiments.

Our simulated robot agent and its environment is a standard design as used in many experiments

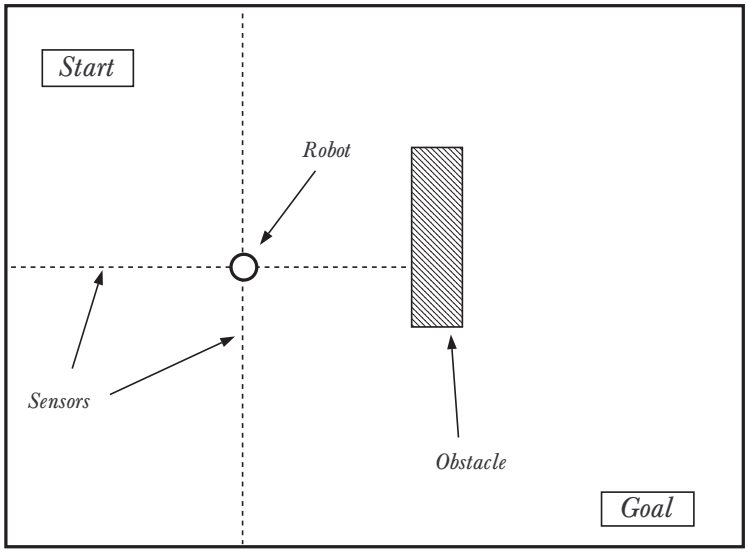

Figure 3: The Environment Used to test Agent Performance

on mobile robot control and navigation. The robot environment is two-dimensional and consists of a rectangular room that can contain simple objects. Figure 3 shows the configuration of the robot world, (of size 400 by 400 units). The robot is represented as a small circle and is equipped with sensors that measure distance (shown as dotted lines) to the nearest object in orthogonal directions: up, left, down, and right. There are two robot outputs; small movement steps in the x and y dimensions, and these are executed in steps called *timeslices*. The defined aim of the robotic agent is simply to move from a given start position to reach a goal position. A real environment can be "noisy" in that the sensors may have inaccuracies and the actions may not be perfectly executed; such errors and uncertainties are modelled in the simulation by adding amounts of disturbance to both the sensor values and the movements executed by the simulator.

The agents initially have no knowledge of their actual position or any details of the environment and can only receive sensory signals and send out motor commands. From this they must build up their own understanding of their robot world. A major problem for any such agent is in knowing whether a change in sensory data is due to noise or is an actual feature of an object in the environment.

A key feature of this particular simulation arrangement is that the behaviour and performance of the agents can be fully observed and accurately measured. The agent designer and experimenter has a "God's Eye" view of the world, being able to observe the whole system directly: agent body, agent "mind" and simulated environment. Such an objective point of view is normally unavailable but is extremely valuable when it can be achieved. Thus, because the actual robot world and the agent's internal model are clearly distinct and are both explicitly recorded, they can be precisely compared to assess the degree of difference and hence the extent of error in the agent's knowledge

of the world.

3.1 The Robot Simulator

The Robot Simulator is a software module that simulates the physical aspects of the agents. The simulator models the environment, simulates robot movements and generates simulated sensory data. It keeps track of the robot's *actual* position and moves it in accordance with motor commands.

At the start of every movement timeslice, the robot simulator performs geometric calculations to compute the four sensory values corresponding to the number of units between the robot and the nearest surface in the up, left, down, and right directions. The simulator also generates robot moves in the x, y space by calculating increments for each direction: Δ_x and Δ_y. These are used to update the robot's position from a previous timeslice $t_{(c-1)}$ to a new location at t_c and are based on the distance moved during a timeslice for a typical robot operating speed (5 units). The increments are reduced appropriately when the movement of the robot would bring it into contact with an obstacle.

The implementation language was Prolog and an object-oriented design was employed, using the SICStus Objects library [1].

3.2 Agent-Specific Implementation Details

As all the agents operate in the same identical environment there are some common functions needed for basic bookkeeping. These deal with the recording of the simulated environment and its graphics display. Also each agent requires storage facilities so that they can build up their own internal models of their environment. Notice that these models of the world are quite separate from the *real* world as modelled by the simulator. In particular, the agents experience obstacles as *surfaces*, not as 2D rectangular objects. This is because, as the agents move around, the sensory values fall into distinct sequences or clusters with occasional abrupt changes to new cluster patterns. These correspond, respectively, to the robot following one "wall" and then moving into view of another one. Consequently, the structures that naturally arise in an agent model are surface representations.

To allow the agents to be able to evaluate the accuracy of their world model, they are able to compare actual sensor readings with readings they would expect to detect by computing "expected" readings from the data they have stored in their world models.

The system operates in a cyclic manner using fixed size timeslices. During each timeslice, the agents perform the sequence: data input, knowledge base modification, and movement generation. These stages are implemented differently according to their theoretical requirements.

3.2.1 Agent SA

The data input stage for agent **SA** is very simple; data are read from the sensors and placed in the relevant data structure for basic beliefs, B_K. Then a forward chaining mechanism is used to infer any consequences D_K from the basic beliefs. A list of relations, in the form of rules, are tested against the basic beliefs and if the preconditions of a relation are met, the relation is applied. This process is repeated until every rule has been tested and no more changes can be made.

Most of the rules used by agent **SA** result in the modification of data. Any inconsistencies between newly derived knowledge and the existing map are translated directly into changes to agent **SA**'s internal map. Agent **SA** has a problem when translating a series of discrete sensor readings into a continuous surface. If agent **SA** finds that its sensor readings do not match its world model, there are two options: it can either create a new surface, or it can modify an existing surface. Surface creation occurs when a sensor reading is less than the agent expects it to be, indicating that a previously undetected surface exists in between the robot and some other feature. However, when a sensor reading is greater than the agent expects it to be, surface modification occurs and the length of the recently experienced surface is shortened, i.e. its end is assumed to have been reached.

In summary, the only technique which agent **SA** is able to use to interpret the environment is the forward chaining of rules based on sensor data. The structure of the rule set allows the forward chaining process to create surfaces based on sensor data and guide the robotic agent to its goal.

3.2.2 Agent WA

This agent is a variation on agent **SA** and in nearly all respects is similar to **SA**. The important difference is **WA**'s use of a suggestion buffer after the forward chaining process and before the editing of map beliefs.

The suggestion buffer is a data structure that stores proposed changes to the agent's map, and, depending on the amount of support received for each suggestion over a variable number of timeslices, either carries out the suggestion by modifying agent **WA**'s internal representation, or ignores the suggestion. A suggestion takes the form of details of a new surface S that originated at timeslice T_x. At the end of forward chaining, all the suggested changes to the map will have been gathered. Then agent **WA** searches through the list of suggested changes to identify which are similar enough to each other that they may be taken to represent *the same* action. This is done by measuring the similarity of every entry in the suggestion list to every other entry, within a pre-determined range of timeslices. The similarities of different actions are measured in terms of the type of action recommended and the new surface co-ordinates which are being suggested. This is a surprisingly complex task as different suggestions made over different timeslices may be suggesting the same changes to the map. The similarity testing must thus be able to compensate for the changes which would occur to the suggestions as the robot moves around the environment.

Suggestions are filtered by selecting, first, all suggestions which lie within a given timeslice window (three timeslices were used for most of the experiments), then similarity of commands and directions was used to eliminate different operations.

The next step is to compensate for variation caused by the movement of the robot. This involves finding the position of the robot when the suggested change was first mooted, and calculating what the suggested modification to a surface *would* have been, *if* the robot had been at the same position as it was when the first suggestion being considered was made. Fortunately, all the information required for these calculations is available from the agent's model.

Notice that simple averaging of the suggested actions will not yield a sensible change to the map. The suggestion buffer code must derive the direction of travel of the robot, and, depending on the direction of travel, use either the maximum or minimum values suggested for the suggested surfaces.

If there is no noise in the environment, this technique will yield identical suggestions originating from successive timeslices, if the suggestions are indeed referring to the same obstacle. If there is noise in the environment, further adjustments have to be made to the position of the surfaces under consideration to compensate for this. These adjustments, which will only involve the position of the surface, can be made by simple averaging of the different positions in the relevant suggestions, as long as the other values in the suggestion are similar enough according to the criteria described above. The suggestion buffer finally assesses the strength of each suggestion, and implements the most strongly supported suggestion, provided its strength exceeds a pre-determined value.

In summary, the suggestion buffer for agent **WA** is simple in concept but the code required to implement its functionality is surprisingly complex.

3.2.3 Agent SH

Agent **SH**, representing an agent based as much as possible on holism, requires extra processes to determine the meaning of every new belief in the context of the whole set of current agent's beliefs. This makes the implementation of agent **SH** considerably more complex than the previous agents.

The mechanism used to keep track of different belief possibilities during processing is based on the idea of interpretations. Agent **SH** stores *every* piece of information within an explicit *interpretation*, which is identified by means of a unique integer (the base interpretation being interpretation 0).

Agent **SH** does not simply read sensory data straight into the relevant belief structures. Instead, such data must be checked against internal knowledge for the derivation of corrected or acceptable data. The process used for this, called backward chaining, forces the system to use many different pieces of data to arrive at sensor values. Another major feature of agent **SH** is the use of meta-level beliefs in the form of explanations to provide *explanation-guided backward chaining.*

The backward chaining process is guided by its attempt to find a value for the "full-explanation" object in the current timeslice. Figure 4 shows the method used to derive sensor data. First, the

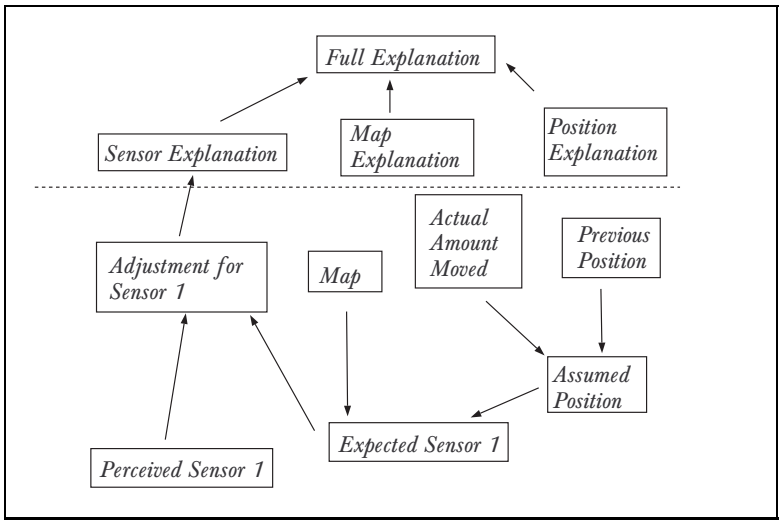

Figure 4: The Relational Structure of Agent **SH**

position at the previous timeslice is combined with the amount by which the robot moved to arrive at an estimation of the robot's current position. Using the current state of the map, this is then used to derive expected sensor readings. Any deviation between the expected sensor readings and the actual sensor readings is temporarily assumed to be noise, and is recorded for that sensor. However, the agent should not simply attribute every deviation between its internal representation and its sensor readings to sensor noise, and so this problem was addressed using a system of *constraints*.

Ontological definitions are maintained for every concept used by the agents. As mentioned before, an object-oriented design has been used for the implementation, and all of the agents' beliefs concerning the external world are represented as distinct but related objects. This also includes information concerning any constraints that are placed on objects, as well as explanation-based information used to guide the resolution of violated constraints. Rather than store every constraint explicitly within each object, constraints are stored within the ontology in a few high level objects, and are *propagated* to lower level objects when appropriate.

Once the backward chaining process has been completed and constraints have been propagated to the relevant objects, the task of finding which constraints have been violated by a given interpretation is relatively straightforward. This is done by obtaining a complete list of objects within the interpretation, and, for each object, obtaining a complete list of constraints which are relevant to the object. This list is then filtered to remove any constraints which have not been violated. Repeating this process for every constraint relevant to every object yields a complete list of constraints which have been violated by the current interpretation.

If no constraints have been violated, the interpretation is acceptable and the system proceeds with the modification of its internal model. Otherwise, the system must resolve the list of violated constraints using a process of *explanation*.

Each explanation-level object (see Figure 4) contains an ordered list of possible explanations which may explain the failed constraint within that particular hierarchy. In this list, the most likely failed constraint is a problem with the map, such as a surface being longer or shorter than expected. The next explanation represents the option that the sensors may have been affected by noise, and the final entry represents the possible explanation that the robot's assumed position at the previous timeslice was mistaken.

The situation is further complicated by the fact that an interpretation may suffer from several failed constraints, each of which have their own recovery list. In this case, the system attempts to choose recovery options which appear in as many recovery lists as possible. This minimises the amount of computation in order to resolve the failed constraints. The recovery process generates new interpretations which address the problems of the base interpretation. The nature of the modifications that are carried out on the base interpretation will depend on related high-level explanations, as shown in Table 2. For further details of these processes involved in agent **SH** see [4].

High Level Explanation	**Interpretation Modification**
Sensor error	Store error values in sensor adjustment objects
Map inaccurate	Edit map accordingly
Position inaccurate	Derive new assumed x and y co-ordinates for the relevant timeslices and assert directly into the new interpretation.

Table 2: Modifications that are Applied to the Base Interpretation Depending on Failure Explanation

To summarise, agent **SH** performs a depth-first search of the interpretation space, whereby every interpretation is based on a different *explanation-level* solution to the problems of its parent interpretation. Once the interpretation tree has been fully grown, agent **SH** will have derived a number of different interpretations, each of which represents a different high-level solution to the problems in the base interpretation. The integrity values of each interpretation are then calculated and the interpretation with the highest integrity score is chosen and copied back into the base interpretation; all other created interpretations are deleted. The base interpretation will then contain the best possible explanation of the available sensor data and it is then possible to simply read the movement values from the current timeslice of interpretation 0, and pass these x and y movement commands directly to the simulator to be executed.

3.3 A Comparison of the Implementation of the three Agents

We now briefly compare the approaches taken as seen in the implementations of the agents. Table 3 summarises the main differences.

Agent	**SA**	**WA**	**SH**
Amount of pre-processing	None		Extensive
Role of existing facts	Minimal	Slight	Extensive
Inference mechanism	Forward chaining		Backward chaining
Number of interpretations	One		Many
Beliefs measurement yardstick	Input data		Entire interpretation

Table 3: Comparison of implemented agent features

The data input stages show some major differences. As Table 3 shows, agents **SA** and **WA** perform no pre-processing of sensor data, but assert data from the robot simulator straight into their internal data structures. Agent **SH** makes extensive use of sensor data pre-processing, in that data which does not agree with its existing beliefs will be modified in such a way as to maximise the coherence of the agent's knowledge base. This effect is achieved using the explanation-based backwards chaining technique which ensures that the driving force behind the derivation of agent **SH**'s knowledge base is the task of maximising the integrity of its knowledge base.

The use of existing beliefs is a major difference between the approaches to knowledge base derivation used by the three agents. Agent **SA** makes use of a few existing beliefs, such as the location of surfaces and the dimensions of the map, when deriving its knowledge base. Agent **WA**'s use of existing factual data is slightly greater due to its use of the suggestion buffer. This operates by measuring the extent to which new suggestions agree with existing suggestions for map modification. Suggestions which receive sufficient support from existing beliefs are implemented, while those which do not are discarded. However, the techniques used by agent **SH** allow it to guide the derivation of its knowledge base at a much higher level, such as assuming problems are caused by map or positioning errors. This involves the use of existing factual beliefs, as the measures of centrality and integrity used to decide between competing solutions compare new beliefs with existing ones.

The evaluation stage of the cycle is the stage during which the agents decide which possible beliefs to accept. The approaches to evaluation taken by the agents vary in two major areas. One is the number of independent *Interpretations* used by the agent during the course of the knowledge base derivation process, and the other concerns *Belief Measurement*, i.e., the yardstick used to measure the acceptability of beliefs. Agents **SA** and **WA** use only one interpretation during the knowledge base derivation process because the atomism on which they are based is incompatible

with the view that the value of beliefs can be found by considering collections of beliefs. This effect is achieved by forward chaining the knowledge base on data received from sensors, and by rejecting information which contradicts sensor data. Agent **SH** is unique in that it uses multiple interpretations. Each interpretation represents an independent solution to a failed constraint based on a different high-level explanation of available data. The value of individual beliefs is only determined by evaluating their centrality within a given interpretation. The interpretations are kept separate by prefixing every interpretation-dependent assertion with an integer describing which interpretation the assertion is relevant to. High level solutions are achieved using a combination of explanation-driven backward chaining and constraint propagation. These high-level solutions are then translated into specific low-level actions by parsing the high-level solution with data relating to the specific failed constraint.

4 Experiments and Results

A set of experiments were designed to provide a comparison of the performance of the three agents when operating under a variety of conditions. The conditions for five experiments are summarised in Table 4.

Experiment	**Environmental conditions**
1	No noise or obstacles (Control)
2	No noise, one obstacle (2 surfaces)
3	One obstacle and high sensor noise
4	One obstacle, high sensor noise and high motor noise
5	One obstacle, low sensor noise and low motor noise

Table 4: The Conditions used for the Experiments

For each experiment, results are given in terms of measures of agent performance. We used five quantitative measures to determine the relative performance of the agents: location accuracy (L_{xy}), surface accuracy (S_r, S_u), the number of surfaces added to the agent's internal map (N_s), the number of seconds required to complete the experiment, and the number of timeslices required to complete the experiment.

The positional accuracy, L_{xy}, is defined as the mean of the robot's positional accuracy over all timeslices, that is:

$$L_{xy} = mean\left(\frac{100}{1+d}\right)_{t=1\ldots n}$$

where d is the Euclidean distance between the robot's actual position (as given by the simulator)

and the position it believes it to be in (as recorded in its internal model). This returns a percentage, where 100% indicates the agent's beliefs concerning its location are completely accurate. The measure S_r concerns the cases where the agent falsely believes a surface exists, while S_u concerns the cases in which the agent fails to identify a surface. A value of 100% means there are no false reports of surfaces or no unreported surfaces, respectively. All statistics given in the results represent the means from at least three separate runs.

4.1 Experiment 1

This experiment was designed as a control case against which to compare the other experiments. Table 5 summarises the performance in driving the robot from the start to the goal position. Figure 5 shows the path taken by all the agents to reach the goal position, and, as seen, all the agents succeeded in reaching the goal position with 100% accuracy. The only significant performance difference was that agent **SH** took 2.7 times longer to complete the task than agents **SA** and **WA**.

Category		**Agent**		
Category	**Sub-Category**	**SA**	**WA**	**SH**
Accuracy	L_{xy} (%)	100.00	100.00	100.00
Execution Time	Seconds	74.54	74.19	200.90
	Timeslices	57	57	57

Table 5: Results from Experiment 1

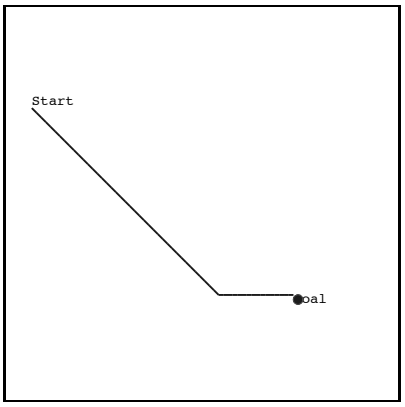

Figure 5: The Path Taken by all Agents in Experiment 1

The performance of agents **SA** and **WA** were virtually identical, showing that the suggestion buffering used by agent **WA** causes no overhead when it is not used. As expected, the coherence-based algorithms used by agent **SH** are more computationally expensive than the algorithms used by agents **SA** and **WA**. However, we see this is the case *even when* the contradiction resolving facilities of these algorithms are not used.

4.2 Experiment 2

In this experiment a single obstacle was placed in the robot's environment, between the start and goal positions. The purpose was to test performance when, unlike in experiment 1, the internal models had to be altered in order to complete the task. As the agents had no prior knowledge of the obstacle they had to use the sensor readings to infer its location and find a path around it.

Category		**Agent**		
Category	**Sub-Category**	**SA**	**WA**	**SH**
Accuracy	L_{xy} (%)	100.00	100.00	100.00
	S_r (%)	100.00	100.00	100.00
	S_u (%)	100.00	100.00	100.00
	N_s	2	2	2
Execution Time	Seconds	86.02	92.66	516.56
	Timeslices	64	65	64

Table 6: Results from Experiment 2

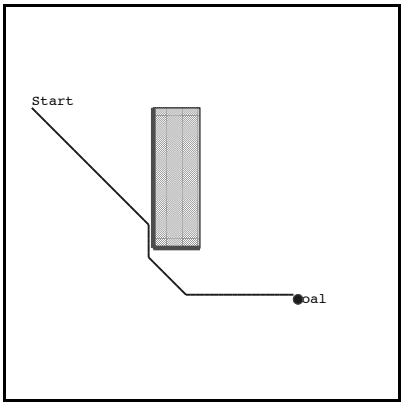

Figure 6: The Path used by Agent **SA** in Experiment 2

The results for experiment 2 are summarised in Table 6. All agents were successful. Figure 6 shows the path used by agent **SA** to reach the goal position. Note that the obstacle is shown on the map only for the purposes of comparison — the agents do not have access to the obstacle's real position. Thick lines in the figures represent the position of surfaces as perceived by the agents, i.e. as created in the agents' internal models.

We see that increased complexity in the environment translates into increased execution times. Agent **SA** took 15% longer to complete experiment 2 than experiment 1. This compares with 25% more time for agent **WA** and 157% for agent **SH**. Thus, agent **SH** is the most sensitive to increased complexity in the environment.

Regarding execution time, agent **SA** recorded the fastest time. This is the expected result, as

agent **SA** had less processing to do than agents **WA** or **SH**. Agent **SH** took 6 times longer to complete the task than agent **SA**. This shows that the algorithms used by agent **SH** are significantly more expensive than those used by agents **SA** and **WA** when the task involves changes to the agent's internal model.

4.3 Experiment 3

In order to explore the effects of imperfect input data, disturbance noise was added to the readings produced from the robot sensors. The implementation allowed the addition of an absolute amount of generated noise, controlled by two parameters: P_n which is the probability that a particular piece of data will be affected by noise (where $0 \leq P_n \leq 1$), and Max_n which is the maximum amount of noise which may be added to the data. The standard Normal distribution is used to modulate the noise to realistic levels.

Sensor noise in experiment 3 used $P_n = 0.5$ and $\text{Max}_n = 15$, thus giving a 50/50 chance of the sensor signals having an error bound of ± 15 (in an environment of 400 by 400 units).

The results obtained in experiment 3 are summarised in Table 7. Figures 7, 8 and 9 show the final maps produced by the three agents. As before, dark lines and dots on the maps represent surfaces which the agents have added to their world models.

Category		**Agent**		
Category	**Sub-Category**	**SA**	**WA**	**SH**
Accuracy	L_{xy} (%)	100.00	100.00	100.00
	S_r (%)	12.59	58.08	100.00
	S_u (%)	6.89	46.00	100.00
	N_s	80	3	2
Execution Time	Seconds	159.24	102.90	514.24
	Timeslices	64	65	64

Table 7: Results from Experiment 3

As would be expected, all agents achieved 100% accuracy in their location, but the results for obstacle accuracy show considerable variation. Agent **SA** uses every piece of sensor data directly in its world model, so noisy sensor data leads to model inaccuracies. This is seen in the considerably lower obstacle accuracy scores for **SA** and the average of 80 surfaces added to its map on each run. Agent **WA** was much more accurate, showing the effectiveness of the suggestion buffer as a noise reduction mechanism. However, agent **SH** achieved near-perfect accuracy in experiment 3 (values were rounded to 2 decimal places). Nevertheless, the price of this increased accuracy comes in the form of increased computational expense.

Agent **SA** took around 50% more time to complete this task than agent **WA**. This is initially surprising, but is explained by the time for **SA** to complete a timeslice increasing in direct propor-

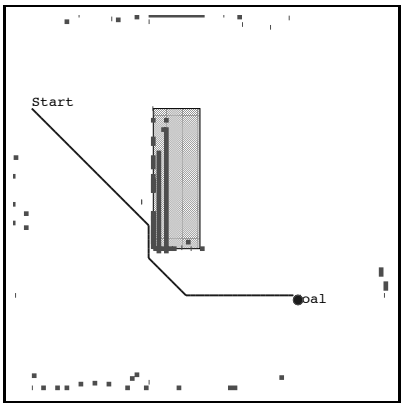

Figure 7: The Final World Model of Agent **SA** from Experiment 3

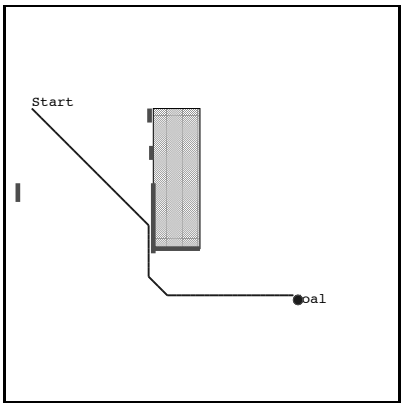

Figure 8: The Final World Model of Agent **WA** from Experiment 3

tion to the number of surfaces in its internal model. Due to the complexity of the coherence-based algorithms agent **SH** took 3 to 5 times longer than the other agents. However, the level of sensor noise used does not affect agent **SH**'s execution time, as this was very similar to the time it took to complete experiment 2.

4.4 Experiment 4

This experiment was similar experiment 3, except that both sensor and movement commands were subject to noise. The sensor noise was as before: $P_n = 0.5$ and $\text{Max}_n = 15$, while the motor noise had $P_n = 0.5$ and $\text{Max}_n = 10$. The simulator might now move the robot an erroneous amount and so this means the robot's actual position might differ from the position it believes it to be in.

The results obtained in experiment 4 are summarised in Table 8 and figures 10, 11, and 12 show the final maps of the agents after sample runs. The dotted thin trail represents the robot's beliefs concerning the path it took to the goal, while the thick trail represents the path it actually took. Similarly, the solid black circle represents the actual position of the robot, while the hollow outlined circle represents the position where it believed it to be.

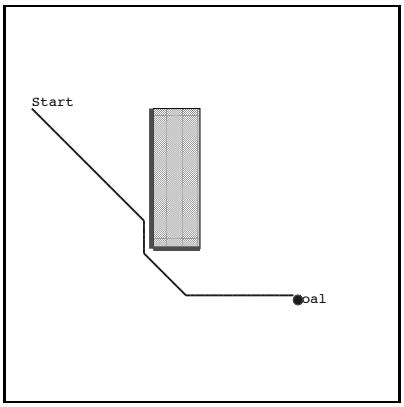

Figure 9: The Final World Model of Agent **SH** from Experiment 3

Category		**Agent**		
Category	**Sub-Category**	**SA**	**WA**	**SH**
Accuracy	L_{xy} (%)	7.61	8.57	12.77
	S_r (%)	3.86	7.11	9.46
	S_u (%)	2.88	4.61	17.25
	N_s	118	12	4
Execution Time	Seconds	225.77	103.18	6351.75
	Timeslices	64	65	72

Table 8: Results from Experiment 4

Figure 10 shows that the difference between actual position and believed positions caused the agent to form an erroneous model of the external world. Despite the sensor noise, one can see how the position of surfaces along the left hand edge of the environment follow a similar pattern to the actual path of the robot. This also explains why agent **SA** consistently placed surfaces too far to the right, meaning that they ended up inside the obstacle's actual position. Indeed, in this experiment, agent **SA** was not able to detect the *actual* position of the obstacle. This is why the path agent **SA** believed it took passes through the actual position of the obstacle.

Figure 11 shows that agent **WA** was better equipped to operate in a noisy environment than agent **SA**. However, like agent **SA**, agent **WA** did not actually succeed in reaching its goal position.

Figure 12 shows that agent **SH** handled the noisy environment reasonably well, as the agent adjusted its beliefs concerning its position on several occasions. Despite the high cost in terms of execution time, agent **SH** was the only agent that actually succeeded in guiding itself from the start to the goal position.

As the task faced by the agents in experiment 4 was a severe one, it is not surprising that all the agents performed worse in experiment 4 than they did in any other experiment. The fact that agent **SA** performed poorly shows that, as expected, an AKB based on the principles of strong

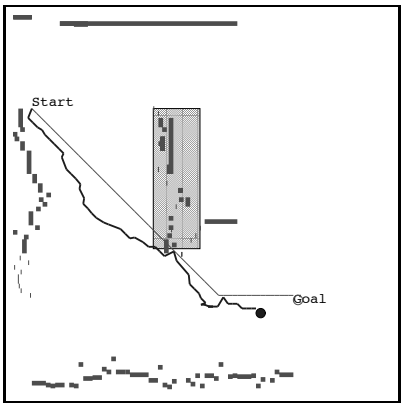

Figure 10: The Final World Model of Agent **SA** from Experiment 4

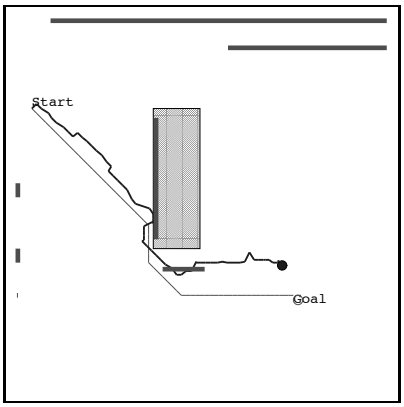

Figure 11: The Final World Model of Agent **WA** from Experiment 4

atomism is not suitable for use in a noisy environment. Agent **SA** added many more surfaces to its map than in experiment 3 and this accounts for the increased processing time. The difference between the performance of agent **WA** in experiments 3 and 4 is a reflection of the fact that agent **WA** was not able to cope with motor noise. However, the maps produced by agent **WA** were more accurate than those produced by agent **SA** and agent **WA** gave similar times as for experiment 3. The fact that agent **SH** required, on average, 8 more timeslices than agent **SA** is due to agent **SH** performing positional adjustments, which agent **SA** was unable to perform. Also agent **SH** required substantially more time to complete the task, i.e. over 60 times the time taken by agent **WA**. This was because agent **SH** was forced to attempt to resolve contradictions at every timeslice and the time taken to do this increased with each timeslice.

4.5 Experiment 5

Experiment 5 was the same as experiment 4 except that the values of sensor and motor noise were both reduced. The reason for this was to reduce the level of disturbance considerably so that a more typical operating result could be obtained.

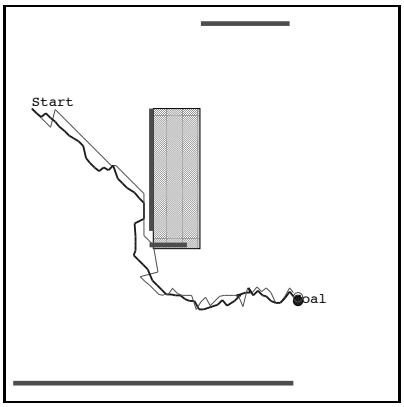

Figure 12: The Final World Model of Agent **SH** from Experiment 4

The results obtained in experiment 5 are summarised in Table 9. Agent **SA** produced better location results but was similar in other respects. Agent **WA** improved its results on surfaces but still produced too many. Agent **SH** gave the most improved performance with all results being considerably better than in experiment 4. The accuracy scores of agent **SH** show that it was much better able to cope with these more realistic levels of noise. It is interesting to note that the average execution times recorded by agents **WA** and **SH** are similar to those recorded by the agents during experiment 3, while agent **SA** recorded an execution time which was comparable to the time it recorded during experiment 4.

Category		**Agent**		
Category	**Sub-Category**	**SA**	**WA**	**SH**
Accuracy	L_{xy} (%)	53.83	51.17	64.00
	S_r (%)	4.13	19.76	66.67
	S_u (%)	2.51	16.24	64.08
	N_s	118	9	2
Execution	Seconds	232.71	111.43	511.66
Time	Timeslices	64	66	72

Table 9: Results from Experiment 5

5 Discussion of Experimental Findings

The experiments have shown several significant differences in behaviour, which we can summarise for the three agents.

Agent SA is much less computationally expensive than agent **SH**, but its performance is poor in noisy environments. The principal advantage of agent **SA** is its relative simplicity. This results in fast execution times, although agent **WA** was faster than agent **SA** in very noisy environments.

When there is no noise agent **SA** is not very sensitive to variations in environmental complexity, but sensor noise causes agent **SA** to (mistakenly) increase the number of surfaces in its internal model. It is even less able to cope with motor noise.

Agent SH achieved the best accuracy results in every experiment, and was also the only agent to successfully complete experiment 4. However, the price for this increased accuracy was increased computational expense as agent **SH** took longer than the other agents in every experiment. This agent was also sensitive to increased environmental complexity, in that its computation time is related to the number of objects in the environment. This means that agent **SH** might not be suitable for use in an environment or application that has a ceiling on response times. However, agent **SH** was not affected by sensor noise and delivered accurate results. It was more affected by motor noise, but when this was at reasonably realistic levels it performed remarkably well; with no increase in execution times and fairly high accuracy.

Agent WA does not possess the theoretical pedigree of agents **SA** and **SH**. However, it consistently achieved better accuracy results than agent **SA** and faster execution times than agent **SH**. This suggests that, from a pragmatic point of view, agents based on less extreme philosophical approaches than those used for agents **SA** and **SH** may yield the most useful systems. Conceptually, the difference between agents **WA** and **SA** is very simple, yet is surprisingly effective. A visual comparison of the final maps reached by these two agents in experiments 3 and 4 shows the effectiveness of not having to translate every change in the environment immediately into a change in an agent's world model. The execution times for agent **WA** were not really affected by either sensor or motor noise. However, its performance, in terms of accuracy, is noticeably better for sensor noise. This is not surprising as agent **WA** was specifically designed to cater for some sensor noise (but not motor noise).

From these results we can make some general observations which bring out some of the design trade-offs available for AKB designers.

5.1 Noisy Environments

If the environment, sensors and actions all have negligible noise, then an atomistic approach would bring benefits of fast and efficient operation. This is because the algorithms for such AKBs can be much simpler and less computationally expensive than those needed for holistic agents.

But if noise was known to be present, or if the degree of noise was unknown, then a holistic approach would give the ability to accommodate incoming data in accordance with the best understanding of the current state of the environment.

5.2 Computational Expense

Most of the extra computational overhead of the holistic approach concerns consistency-maintenance. For this reason, it is expected that the performance of such systems will compare favourably with

that of atomistic systems in noise-free environments. Hence, a system operating in a noise-free environment could incorporate some holistic functionality without a significant loss of performance, while still possessing the ability to deal with inconsistencies should they arise.

5.3 Fast Response Times

In situations with real-time constraints, where it would be desirable for an agent to react as soon as possible to a given input, the holistic ability to modify sensor data would not be an advantage.

Similarly, there may be situations in which the internal coherence of the system is less important than its ability to provide solutions as quickly as possible. For example, it might be expected that an atomistic agent would not possess the functionality to form a consistent model of a noisy environment. However, as long as the agent has an approximate concept of its position in relation to its target, it will still produce movement commands which will move the robot nearer to its target. While the lack of accuracy will probably result in the atomistic agent requiring more movement commands than a holistic agent, the fact that the agent will be responding more quickly than a holistic agent can mean that the robot reaches its destination in a similar or even faster time.

5.4 A Consistent Overview

Certain applications may attach more importance to the ability to generate several consistent alternative explanations of external events than the ability to respond immediately to external world changes. Such applications would benefit from holistic systems which incorporate the functionality to re-arrange existing beliefs in order to maximise coherence.

6 Philosophical and AI views of the Agent Designs

While agent **WA** achieved better accuracy than agent **SA**, questions can be raised over the extent to which agent **WA** is actually based on the extreme atomism of **PA.** By allowing existing beliefs to take priority over individual pieces of sensor data the emphasis for the agent's knowledge base has moved a small but significant amount towards the holistic end of the spectrum. From an AI point of view, the additional complexity of **WA** is due to the amount of "common sense" reasoning that is required for tasks involving simple spatial and temporal reasoning. From a philosophical point of view, this complexity is due to the fact that pure atomism is a relatively simple concept compared with anything moving towards holism.

The complexity of agent **SH** can be explained from an AI perspective as being due to the fact that the agent is using a constraint satisfaction method for controlling an embedded robot. This is a complex task, and it comes as no surprise that using constraint satisfaction together with backward chaining is more complex than using forward chaining, which is much simpler to implement. From a philosophical point of view, the complexity of agent **SH** is due to the fact that

it is based on radical holism, and, as such, the meaning of every belief has to be determined by the *whole* set of the agent's beliefs.

We have seen how the most "obvious" approach, i.e., that of atomism, where sensory experience, being accepted on face value and stored away, and thus given importance as some kind of objective truth, does not necessarily deliver the best performance. Counter-intuitive approaches can, surprisingly, deliver better or even more appropriate results. The holistic approach can be seen as a form of subjective system where new experience has to fit in with the internal beliefs about the world. Clearly, there are many other possibilities but the message for agent designers must be that common sense approaches are not reliable as the only source of inspiration.

In Section 3.3 we compared the features of the implemented agents. Following our analysis, Table 10 expands and elaborates on the three key processing stages and the functionality they entail. This summarises our implementations and illustrates some of the options open to designers.

7 Conclusions

In a previous paper we showed that philosophical theories were relevant to the AI field of autonomous agent design [5]. We have now tested this idea by implementing a series of agents with different epistemological backgrounds. The purpose of the implementations was to firmly ground the theories and concepts, so that we can precisely examine any significant differences in their behaviour. This shows how it is possible to design and implement agents *explicitly* based on particular philosophical theories. The agents have exhibited different types of behaviour which reflects the designs upon which they were based, that in turn can be attributed to the differences in their various philosophical positions. Thus, the philosophical basis of a system *makes a difference* to the design and behaviour of that system.

It is rare in AI to find alternative implementations of exactly the same problem, and equally rare to see attempts to repeat experiments through re-implementations. Yet such experiments are necessary if we are to understand fully the implications of the range of design options open to us when building agents and AKBs. This study illustrates the range of design differences that can be explored by taking different epistemological starting points on a common problem.

It is well known that the disciplines of knowledge representation and epistemology are related. Knowledge representation is concerned with the task of representing a portion of the unbounded external world within the bounded world model of a cognitive agent. Epistemology is concerned with the validity of this representational task. It is the same features of the world which cause problems for epistemology and knowledge representation and, therefore, some of the techniques which have been developed by philosophers to address epistemological issues have relevance to AI problems.

However, despite AI's long history of research into knowledge representation, reasoning and inference techniques, applications for cognitive agents are less well advanced. Recent developments

Functional Stage	**Theoretical Concept**	**Approach Based on PA**	**Approach Based on PH**
Data Input	Accuracy	Defined in terms of correspondence with sensor data.	Defined in terms of coherence with existing beliefs.
	Status of model	Internal model defined in terms of external world.	External world defined in terms of internal model.
	Truth values	Beliefs may be either true or false.	Beliefs may be true, false, or unknown.
Derivation (combining new beliefs with existing beliefs)	Theory of justification	**Foundationalism** Knowledge base derived by forward chaining of basic beliefs.	**Coherence** Knowledge base derived by explanation-driven backward chaining.
	Relative importance of beliefs	Asymmetrical. Justification based on non-inferred basic beliefs.	All beliefs are inferred. Justification is symmetrical. All beliefs are revisable.
	Internal model based on	Agent **SA**. Direct consequences of sensor data.	Maximising the coherence of the knowledge base.
		Agent **WA**. Sensor data averaged from multiple timeslices.	
Evaluation (knowledge base revision)	Inconsistency detection	Individual inconsistent beliefs.	Based on violated constraint detection.
	Dealing with inconsistency	Agent **SA**. Label inconsistencies and await more accurate sensor data.	Holistic explanation-based recovery involving multiple independent candidate knowledge bases.
		Agent **WA**. Use of the suggestion buffer to minimise the effects of sensor noise.	
	Solution selection	None	Choose most coherent candidate knowledge base.

Table 10: The Functionality Expressed by the Three Agents

that promise autonomous domestic agents will have to face issues such as bounded models for unbounded environments and the central distinction between self and non-self. Indeed, if they are to eventually support a dialogue with other agents they will also have to create and maintain a model of those agents' beliefs within their own cognitive model. This illustrates how different design options can be important. For example, complex agents that need to reason about other agents may decide to reject incoming sensory information if it conflicts with their experience, even if other agents accept the same information. Assuming that all agents begin life as identical clones (because customisation must be done *in situ*), this situation is more likely to be supported by a holistic design, than an atomistic approach. Thus, the more obvious or common sense assumptions for AKB designs often have the tempting appeal of simplicity, but less intuitive theories may also offer solutions, possibly with much greater benefits.

It is important to recognise that there is substantial disagreement among philosophers concerning which set of theories is correct, and indeed, different philosophers will often espouse subtly different formulations of the same concept. We took two opposing perspectives for our study, but our resulting designs were just one set of concrete realisations of many possible alternatives. Many other interpretations could be produced and these would lead to many variations and design options. Thus, we are not claiming that the adoption of a particular set of philosophical theories will necessarily be associated with unique functional and algorithmic approaches to the design of the agent. Indeed, it is almost inevitable that the personal preferences of the individual researcher will lead to alternative AKB design choices at the functional and hence algorithmic levels.

As computer scientists, AI researchers may feel uneasy with this level of subjectivity. It is our assertion, however, that the process of transforming a theory from the philosophical level to the functional and algorithmic levels *always* occurs in AKB design, albeit implicitly. By making this process explicit, AI researchers will open up their access to the vast assortment of philosophical theories concerning areas which are relevant to AI, such as truth, justification, memory, and perception, which have already been the subject of rigorous philosophical study.

Acknowledgements

We are very grateful to the Department of Computer Science at the University of Wales, Aberystwyth for providing funding for Nick Lacey. Thanks to George Macleod Coghill for discussions and comments on this paper and also to the anonymous referees for their valuable suggestions and improvements.

References

[1] Jonas Almgren, Stefan Andersson, Mats Carlsson, Lena Flood, Seif Haridi, Claes Frisk, Hans Nilsson, and Jan Sundberg. *SICStus Prolog Library Manual.* Swedish Institute of Computer

Science, Kista, Sweden, January 1993.

[2] Susan Haack. *Evidence and Inquiry: Towards Reconstruction in Epistemology.* Blackwell, Oxford, UK, 1997.

[3] J. C. T. Hallam and C. A. Malcolm. Behaviour: Perception, action and intelligence — the view from situated robotics. *Philosophical Transactions of the Royal Society, Series A: Physical Sciences and Engineering*, 349(1689):29–42, 1994.

[4] Nicholas Lacey. *Investigating the Relevance and Application of Epistemological and Metaphysical Theories to Agent Knowledge Bases.* PhD thesis, University of Wales, Aberystwyth, 2000.

[5] Nicholas Lacey and Mark Lee. The epistemological foundations of artificial agents (to appear). *Minds and Machines*, 2003.

[6] Chris Malcolm and Tim Smithers. Symbol grounding via a hybrid architecture in an autonomous assembly system. *Robotics and Autonomous systems*, 6(1):123–144, 1990.

[7] Mary-Anne Williams. Anytime belief revision. In Martha E. Pollack, editor, *proceedings of the 15th international joint conference on artificial intelligence (IJCAI 97)*, pages 74–79, Nagoya, Japan, 1997.